18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

URREF for Veracity Assessment in Query-Based Information Fusion Systems

Erik Blasch
Air Force Research Laboratory
Rome, NY, 13441
erik.blasch.1@us.af.mil

Alex Aved
Air Force Research Laboratory
Rome, NY, 13441
alexander.aved@us.af.mil

***Abstract*— The *uncertainty representation and reasoning evaluation framework (URREF)* ontology discusses an organization, categorization, taxonomy and definitions of uncertainty. Uncertainty can be either subjective or objective pending its source. For example, the information's origin can affect the evaluation, processing, and results from information fusion systems. In this paper we explore the coordination of a user and machine, and its effects on the ontology; our focus is on query-based systems. A user requesting information provides refinement of the analysis via queries which are typically semantic in nature. A query that is continuously refined reduces uncertainty and the information can be used in the ontology as part of the source, evaluation, and information quality. For example, a user receiving a response to a query can determine if the response meets the quality, evaluation, and source criteria. The user response (i.e. relevance feedback) to "did this answer your question," can subsequently alter the uncertainty values. Together with the URREF ontology, query-based information fusion systems can work synergistically to reduce decision uncertainty.

Keywords: *Query-based systems, URREF, Level 5 User Refinement, High-Level Information Fusion, Semantic Labels, Multimedia Indexing, Live Video Computing DataBase Management System*

I. INTRODUCTION

Current developments in cognitive computing, cloud architectures, and distributed access afford people access to a large amount of multimedia information [1]. Multimedia constructs include content (data), entities (features), and scenes (context). Context enhanced information fusion examples include imagery [2], content-based image retrieval (CBIR) [3], tacking [4,5,6] text and tracking [7, 8] user queries [9]. The multiple applications of fusion require resource management [10] to facilitate the ability of the user-defined queries to be determined from the information management system. To enhance multimedia developments, we focused on database designs to increase responsiveness to queries, increased accuracies, and greater situation awareness as context assessment and context management [11].

Early database systems were designed to efficiently manage the storage, retrieval and querying of alphanumeric data [12]. A typical *database management system* (DBMS) implementation supports business applications by persisting application state, resolving queries, and facilitating transactions to mitigate concurrency errors. A *Multi-Media DataBase* (MMDBS) utilizes a traditional DBMS to manage metadata and indices, but also encompasses additional technologies and services to include: video on demand, document management and imaging, spatial data, specialized query languages, relevance feedback. Because multimedia content, and video in particular, can be quite large and its communication bandwidth intensive, MMDBS are often paired with specialized communication frameworks [13].

To enhance multimedia content, we use URREF [14] in the live video database system [15]. URREF has been developed by the International Society of Information Fusion (ISIF) *Evaluation of Technologies for Uncertainty Representation Working Group* (ETURWG) from which multiple discussions have developed over different aspects of uncertainty. The URREF was used for analysis over imagery [16], detection [17], and trust [18]. Inherently, it is the ontology of metrics of uncertainty that can support DBMSs. The key elements we wish to utilize are data quality issues of accuracy, precision, and veracity (as shown in the current categories of the URREF in Fig. 1) within a data handing architecture. While accuracy and prediction have been explored, it is *veracity* that is yet to be agreed upon.

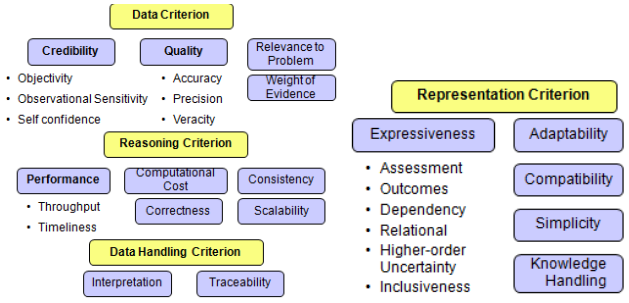

Figure 1 – URREF Categories

Veracity is an element of Big Data, which includes Validity, Volatility, Veracity, Velocity, Variety, Volume, and sometimes Vividness [19]. Typically, velocity, variety, and volume are associated with streaming multimedia along with veracity (Fig. 2). Veracity refers to the biases, noise and abnormality in the data. Thus, multimedia indexing, storage, and retrieval should have meaningful (e.g., high veracity) data to answer queries to problems being analyzed [20]. Precise data reduces user workload [21]. We are interested in measuring the data veracity as determine whether the source data is true. Very few papers discuss veracity data analysis, so this is an attempt to implement for the URREF in an example.

The paper continues with Sect. II discussing context indexing, retrieval and uncertainty analysis (e.g., veracity). Sect. III presents the database design and Sect. IV details the *Live Video Computing* (LVC) *DataBase Management System* (LVDBMS). Sect. V presents results of software coordinated video measurement, and Sect. VI provides conclusions.

II. MULTIMEDIA DATA REPRESENTATION

A. Multimedia Indexing

Multimedia constructs include content (data), entities (features), and scenes (context) that are utilized through indexing and retrieval. Emerging Dynamic Data Driven Application Systems (DDDAS) [22] constructs include multimedia information [23,24,25,26]. Collections of multimedia information can grow to very large sizes, consuming many gigabytes of storage space. In order to utilize multimedia content it must be retrieved; whether the retrieval is to find a movie based upon its title, or one is looking for images, clips of audio or video segments showing a particular subject or class of objects. As an example, consider a table of records in a traditional relational database. Each record in the table can be considered as a point in a multidimensional space [27, 28]. Consider a record for an object-track relation with the following fields: {*object_id, track_id, situation_id, start_date, end_date*}. In this case, records in this table correspond with points in a 5-dimensional space, where three of the dimensions refer to, say, integers (*object_id, track_id* and *situation_id*) and the other two dimensions are of type date-time (i.e. *start_date, end_date*). The DBMS manages the collection of these records and stores. In order to facilitate efficient retrieval of records in the database, ***indexes*** can be created.

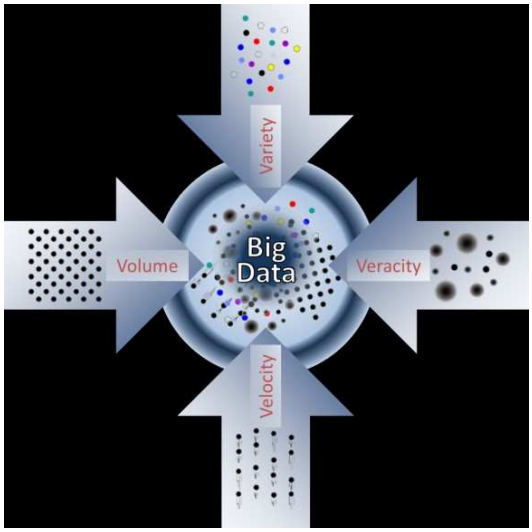

Figure 2 – Big Data (adapted from Claverie-Berge, 2012, [29]).

The index itself is another entity (created and maintained by the DBMS). For example, an index over a table may contain only key values and the locations of records in the corresponding table. By utilizing indexes to resolve queries, less data has to be processed, since the index represents its corresponding record but more concisely. To further enable efficient retrieval, an ordering can be imposed upon the records, either in the primary data file or in the index. However, to accommodate future record operations to the primary data table (e.g. delete, insert, update) it is often more efficient to impose the ordering only on the data in the index files. For numeric fields, the ordering can be based upon the numeric value.

Samet [29] identifies key questions to represent a dataset:

(1) *What* is the type of the data; continuous, discrete?
(2) *When* will operations be performed; e.g. a log file might only have data appended to its end.
(3) *How* should the ordering of the data be applied; should the data in the primary file be ordered, or only the index files? Which attributes should be included in the ordering?
(4) *Which data* be added or removed? Will additional attributes be added in the future? and,
(5) *How much* is the quantity data - will fit into the primary memory of the computer hosting the database, or will disk-resident data access algorithms need to be utilized?

When considering multimedia for browsing and searching, an index is required to make the responsiveness, timeliness and corresponding computation requirements associated with browsing achievable. Some fundamental questions pertaining to multimedia data are *what*, *which* and *how*. At *what* granularity should the item be represented in the index; as a whole or by frame or a clip of frames? *Which* refers to which items should be indexed; such as - should all pixels shown in each frame of video be represented in the index, or only changing pixels? Should the time index of when an object enters be recorded? *How* to index an item pertains to feature selection and storage. Multimedia data indexing is a multifaceted problem, and as such, there is a significant quantity of research [30,31,32, 33].

To illustrate ***multimedia indexing***, consider information that can be extracted from video: visual information (pixels), auditory information, text (text that can be extracted); and metadata (e.g., date, genre, actors). A multitude of semantic properties of the video can be extracted from the metadata pertaining to its content: the type of video, the time period the video covers; major actors who appear, etc. [34, 35]. To index content that is depicted, pattern recognition can be employed; for example, template matching (e.g., Bayes classifier, decision trees, Hidden Markov Models, face and people detection) [36]. The reader is referred to A. Jain [37] for a comprehensive review of pattern recognition techniques. To index videos, they can be decomposed into a series of semantic shots, and each shot can be individually indexed [38, 39]. To index audio data, a number of different techniques can be employed, for example sounds can be analyzed to detect musical instruments or talking [40,41].

B. Multimedia Retrieval

To index multimedia content, first data is decomposed and segmented and features which correspond to points in a multidimensional space are extracted. These feature points represent the associated multimedia content.

Storage of data requires an organization; logically data is organized into buckets and physically the buckets are oriented in ***pages***. Pages (and correspondingly, the buckets containing data points) are stored in files such as a Grid File [42].

Another straightforward technique to organize data in a file is to utilize a ***hash function***. A hash function is a mathematical function to distribute key-value pairs into storage buckets. Given a key, the hash function can suggest which bucket to store the content. In the case that the bucket is at capacity, there are various algorithms that determine how to manage the overflow [43,44, 45].

A ***tree structures*** index supports different data type storage; and the types of queries that will be performed, including point queries, range queries, and window queries. For point data, one can utilize index structures like the Binary Search Trees [46], B-Tree or B^+-Tree [47]. The Space-Partitioning Generalized Search Tree (SP-GiST) index supports input-output messaging when the tree structure is unbalanced [48].

When working with high-dimensional data, one method of data management is to reduce the dimensionality and utilize one of the hierarchical data structures, such as an R-Tree [49]. Alternatively, indices that are not based upon the dimensions of the objects are the distances between them, e.g., SparseMap [50], FastMap, and MetricMap [51].

C. Multimedia Uncertainty Analysis

With multimedia indexing and retrieval, there exists an unknown modeled effect of uncertainty as the data is stored with adherence to the data uncertainty. Using the current definitions from the URREF (Table 1), precision, accuracy, and veracity are elements of the ontology.

Table 1: URREF Data Criterion: Quality

Quality	**Accuracy**	Closeness of agreement between an evaluation subject value and the true value of the quantity or quality being evaluated (adapted from JCGM 200:2008)
	Precision	Closeness of agreement between indications or measured quantity values obtained by replicate measurements on the same or similar evaluation subjects under specified conditions (adapted from JCGM 200:2008)
	Veracity	Measure of the extent to which a source reports what it assesses to be the case.

Veracity, as described by Lukoianova and Rubin [52], has three main dimensions: 1) objectivity/subjectivity, 2) truthfulness/deception, and 3) credibility/implausibility. Information quality requires uncertainty management. Content management includes quantifying the levels of ***content objectivity***, ***truthfulness***, and ***credibility*** (OTC) [52]. Lukoianova and Rubin suggest the use of mutual information over normalization of the OTC for a veracity index although no results are presented. In what follows, we describe the LVDBMS system and veracity within the URREF.

Table 2. Comparison of LVC and traditional DBMS concepts.

Concept	**LVC**	**DBMS**
Storage	Camera	Hard Drive
Relation	Video stream	Record
Data unit	Video frame	Tuple
Data granularity	Object	Attribute
Query language	LVQL	SQL

III. Database Systems for Context Analysis

Traditional DBMSs orient data in *tables* containing records. Each record in a table has a common attribute structure, illustrated in the right side of Fig. 3. ***Live-Video Computing*** (LVC) is a stream-oriented paradigm that operates over streaming data as input. A comparison of concepts between traditional database computing and LVC is presented in Table 2 and graphically illustrated in Fig. 3. ***Live Video Query Language*** (*LVQL*) is the query language of the LVC implementation to specify events in terms of spatio-temporal observations and correlations of objects in video streams.

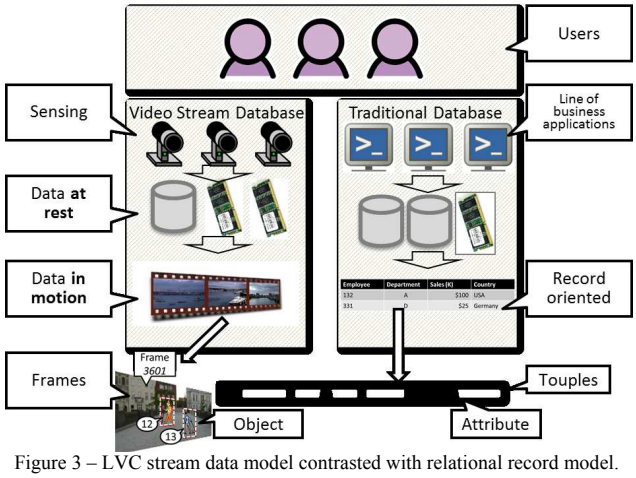

Figure 3 – LVC stream data model contrasted with relational record model.

LVC is the theoretical framework upon which the LVDBMS prototype (e.g., depicted in Fig. 4) system is based. Traditional video stream processing applications are designed specifically to solve a particular problem, termed *siloed*. If context data developed from one application needs to be combined for auditing, reporting or other purposes, additional software (middleware) must be purchased and interfaced with siloed applications. For siloed systems, middleware must be installed and configured on a case-by-case bases and "adapters" for each application must be configured or developed to provide application-specific interfaces to the middleware.

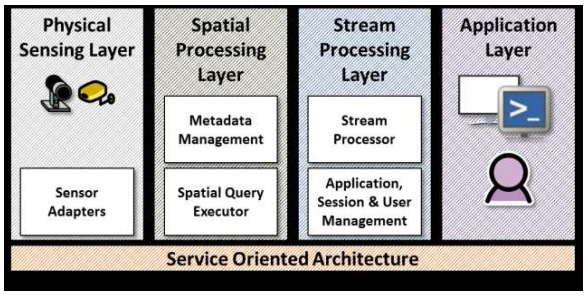

Figure 4 – Logical relationship among components of the LVDBMS prototype and major components of the framework encapsulated in each tier.

The middleware performs an *extract, transform and load* (*ETL*) process to transform data received from the application-specific adapters into a common data format that is amenable to further processing. The result is additional middleware software that must be purchased and maintained and also staff resources to install, configure, maintain and upgrade, as appropriate. Fig. 5 illustrates the common repository approach with middleware, which centralizes some processing and business logic but does not address redundancy and resource sharing and contention issues with the physical sensors.

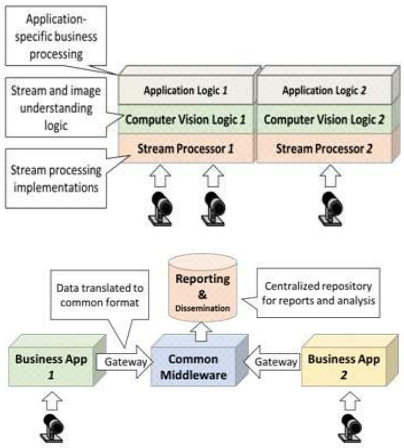

Figure 5 – Applications built as information silos utilizing middleware and application-specific data adapters.

Libraries such as OpenCV [53] and *Integrated Performance Primitives* (*IPP*) [54] are commonly used by programmers when developing repository applications, to provide basic data-handing functionality. OpenCV provides a comprehensive assortment of image processing and data management routines and data structures. For example, the IPP library provides functions and associated data structures that are specifically tuned to take advantage of features provided by modern multicore processors such as parallel data processing instructions. However these common libraries provide low-level functionality that programmers use as conveniences; and do not generally provide out-of-the-box high-level application functionality. (For example, OpenCV routines could be used to read in frames from a camera, and other routines would need to be called in the proper order with the proper parameters and settings in order to interpret imagery depicted in the frames.)

The LVC approach leverages (1) a common video processing software infrastructure to provide a common programmable interface to clients, (2) a shareable pool of camera resources to share context data with the goal is to create an ecosystem for collaboration and information sharing, and (3) searchable database to allow users to draw new insights that are not possible with siloed information frameworks (Fig. 4). The LVC approach facilitates rapid application development by allowing application architects and software developers to focus their time and resources on the business problem at hand, rather than implementation issues pertaining to computer vision and stream processing implementations.

IV. LVDBMS For Context Analysis

The components of the LVDBMS are logically grouped into four tiers, illustrated in Fig. 6. Each tier defines one or more web service interfaces to facilitate communication between the layers. The four layers include:

- The *camera layer* encompasses cameras and their corresponding adapters. Camera adapters are conceptually similar to device drivers in computer systems, allowing for disparate camera device hardware to connect with a standard LVDBMS interface.
- The *spatial processing layer* processes the metadata and video streams from the camera adapters and passes results to the stream processing layer. A host in this layer communicates with multiple camera adapters, but a camera adapter communicates with only a single spatial processing layer host.
- The *stream processing layer* receives subquery evaluation streams from spatial processing layer hosts and computes final query results for delivery to clients. As this interfaces with end users and applications (i.e. the client layer), it contains logic for managing authentication, connections and session state with LVDBMS clients.
- The *client layer* encompasses LVDBMS end users and client applications. Clients authenticate and interact with the LVDBMS by browsing the catalog of cameras, submitting queries and receiving query results.

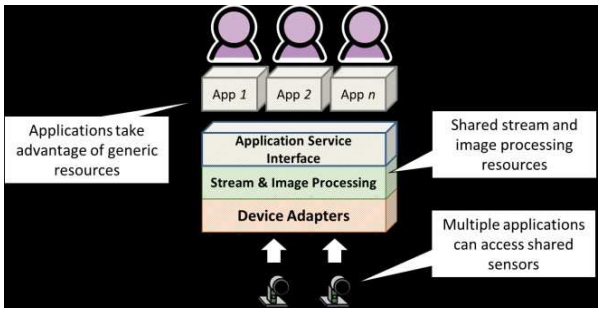

Figure 6 – LVC approach showing common core software stack and shared sensor resources, all aligned for multi-use and multi-workload processing.

The LVDBMS illustration is depicted in Fig. 6 and is refined in Fig. 7, which illustrates how a query flows down through the LVDBMS architecture, and then how data and query results flow back up through the layers and back to the client.

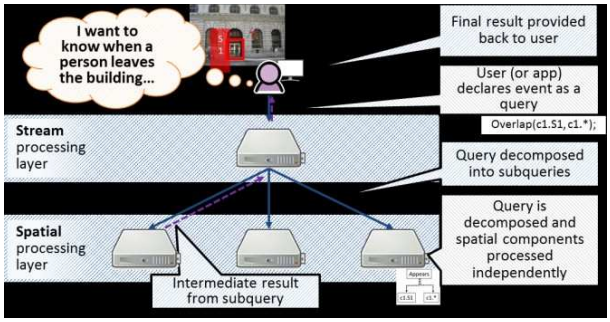

Figure 7 – LVDBMS query and subsequent decomposition.

A ***query*** is posed by an end user or client application to the LVDBMS. The initial query is submitted to the stream processing layer host to which the client is connected. The stream processing layer host maintains metadata pertaining to available spatial processing layer hosts (also referred to as camera servers, as they interface with cameras via their adapters and perform processing) and their associated cameras. With metadata information, the stream processing layer host translates a query into one or more subqueries. Each subquery corresponds to a particular camera server host, where it will be sent for evaluation. Camera adapters process imagery from camera sensors and translate it into a stream of

images and corresponding metadata, which is sent to its respective camera server. Metadata associated with each video frame from the camera adapter includes information pertaining to the frame itself (i.e., timestamp, sequence number, etc.) and to objects observed within the frame and segmented out by the camera adapter (i.e. object identifier, a bounding box identifying the location of the object within the frame). Subqueries evaluate LVQL expressions over video streams (specifically, intersecting video streams specified by the query and video streams managed by a particular camera server to which the subquery was sent) and stream subquery evaluation results back to the respective stream processing layer host. The stream processing layer host receives one or more intermediate results for each evaluation time step and computes a final query result (for the particular point in time), which is then delivered back to the end user or client application. The query information is based on the data model.

A. LVC-DMBS Data Model

LVC, and correspondingly the LVDBMS, is concerned with computation over video streams. As such, the event and data models revolve around objects that are observable by imaging sensors and depicted in temporally oriented frames in the video streams that emanate from these sensors. Therefore, it follows that an event (i.e., a *simple event*) is defined to be occurrence of an action that may be observed by one (or more) cameras and represented in frame data in corresponding data streams. We note that in this work, the terms video stream and camera stream are used interchangeably, as are enabling hardware device terms such as camera and imaging sensor.

With a LVDBMS client, *events* may be specified in LVQL by using a combination of spatial and temporal components, or operators. Thus, a user can leverage the Live-Video Query Language (LVQL) to specify a *complex event* in terms of simple temporally-related events. For example, a simple event could be a person (or object) appearing in a scene. A complex event, or *activity*, relates simple events with temporal operators [55]. For example, a complex event could be defined as a person first appearing in a scene and then, within some threshold of time, moving in front of a car.

A ***spatiotemporal query*** is formulated in LVQL. This query specification defines which video streams will be monitored for the occurrence of an event. That is, if the query specifies that a particular video stream will be monitored for the appearance of an object, if an object subsequently appears in a different video stream, there will be no impact upon the query result. An object is a fundamental component of an event specification. As indicated in Table 3, there are two basic types of objects that are recognized: *dynamic objects* are detected automatically by the image processing software, and *static objects* are indicated by users of the system. The third class of objects is *cross-camera* dynamic objects. These are dynamic objects that were first recognized in one video stream and subsequently recognized in a second stream. The inclusion of the cross-camera object class simplifies the expression of queries that define events correlating objects that appear in multiple video streams. Note that in each respective stream these objects also qualify as dynamic objects, Fig. 8 illustrates the interrelationships among the various objects graphically.

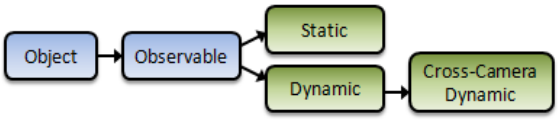

Figure 8 – Illustration of ontology interrelationships among LVDBMS objects.

Table 3: Comparison classes of LVC objects

Object Class	Description
Static	Objects of this class are defined by the user and do not move within the scene. For example, a static object may be defined (drawn) over a window or door for subsequent use in a query.
Dynamic	Salient objects that are detected automatically within a video stream. A model of the scene background is maintained and as an object passes through the scene, its appearance is distinguished from the background. If its size is beyond a threshold it is segmented, assigned a unique identifier and tracked
Cross-Camera Dynamic	Objects detected in one video stream and subsequently matched to an object in a second video stream are classified as cross-camera dynamic objects

Another view of the data flow in the LVDBMS is presented as an example in Fig. 9. Starting from the left, a camera observes a scene of a sidewalk and building. Within the scene objects are observed, including a door and a pedestrian walking assigned identifiers and tracked within their respective video streams. (Note that the door is manually identified by an operator.)

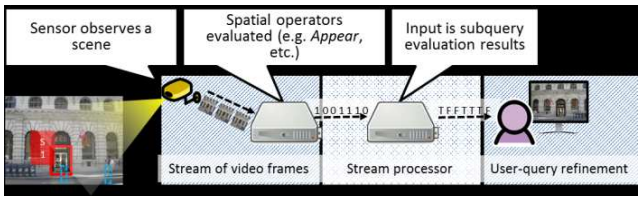

Figure 9 –LVDBMS data flow, from sensor to frames to query result.

As an example, Fig. 9, the scene is segmented and objects are tracked and sent to the server in the spatial processing layer. The camera server uses the metadata received from the camera adapter to process the spatial operators and send the stream of results to the stream processing server residing in the stream processing layer. The final query result is streamed to the user from the stream processing layer host.

B. LVC-DMBS Query Language

LVQL is the query language of the LVDBMS that enables contextual acquisition of information. Analysts and programmers leverage the query language in order to develop applications that interact with video streams and (in the future) other modalities of sensors. As such, the programmers and application designers need only know the details of the query language, and do not need to spend time developing stream processing algorithms or low-level details of the LVDBMS. LVQL permits for the specification of an event and a corresponding action to be defined over a video stream (or a set of video streams). LVQL a *declarative language*, meaning that the user defines a logical event specification and not the particular flow of control or algorithms that will be executed to determine the query result. An LVQL expression specifies a

spatio-temporal event, and an action that is to be triggered when the event is recognized. The basic form of a query (specifically, an *ActionEvent*) is as follows:

ACTION *UserSpecifiedAction*
ON EVENT *EventSpecification*

which signifies that an action *UserSpecifiedAction* corresponds with *EventSpecification* and will be executed the first time a query evaluation result of *true* is returned. *EventSpecification* is an event specification that is generated by a context free grammar which consists of a set of rules, or productions, which can be utilized to express (describe) an event. A simplified set of LVQL productions is presented in Fig. 10 items shown in light blue represent tokens recognized by the language.

```
Lvql := ActionEvent | VDL
ActionEvent := [ action UserSpecifiedAction ] on
       EventSpecification [ period ]
EventSpecification := NotSpTmplEvent ( BooleanOperator
       NotSpTmplEvent )
NotSpTmplEvent := [ not ] SpatialTemporalEvent
SpatialTemporalEvent := CompositSpatialEvent |
       CompositTemporalEvent
CompositSpatialEvent := appear | north | northwest | inside
       | meet | ...
CompositTemporalEvent := before | meets
BooleanOperator := and | or | not BooleanOperator
VDL := VCmdType view ViewIdentifier over VStreamIdent
       [ set VPrivFilter ]
VCmdType := create | update | delete
VTargetStmt := target eq ( querytargets |  nonquerytargets
       | previouslymasked | none )
VTmpScpStmt :=  temporalscope eq ( querynonactive |
       queryactive |    permanent | none )
VObjScpStmt := objectscope eq ( static | dynamic |
       crosscameradynamic | none )
VStreamIdent := ( Cameraidentifier | ViewIdentifier )
Cameraidentifier := camIdent
ViewIdentifier := viewIdent
```

Figure 10 –LVQL grammar (omitting parameters such as thresholds).

Fig. 10 shows the LVQL syntax which consists of either an *Action Event* or a *View Definition Language* (*VDL*) production. In the case of an *Action Event*, which specifies a query, the event definition must contain a spatial operator (e.g. *Appear*, *North*, *Meet*, etc.) Events in LVQL are expressed in terms of spatial and temporal operators and Boolean logic. The simplest event that can be expressed is the appearance of an object in a video stream by using the *Appear()* operator. The *Appear()* operator accepts two arguments (i.e., operands) where the first operand specifies the video stream, the object class (and possibly filter criteria) that the operator will be applied to, and the second is a threshold. (All spatial operators accept a threshold argument.) The threshold for the *Appear()* operator specifies the minimum size of an object that will satisfy the appearance condition, in terms of the area of the *minimum bounding rectangle* (*MBR*) that contains the object. For example, *Appear(s1.*, 200)* will return true each time it is evaluated if a dynamic object with an MBR of area greater than or equal to 200 is observed in the current video frame. In the case of a spatial operator such as *West()*, three arguments are accepted; the first two correspond to objects in the video stream, and the third is again a threshold. *West()* returns true if the object specified by the first operand is to the left of the object specified by the second operand, in a stream. The third argument, the *threshold*, specifies the amount of separation between these objects. The ability to ascertain properties of the target resolves context such as geometry, orientation, intensity, etc. for context assessment.

V. RESULTS

The LVQL query language provides an interface between operators (real-time), analysts (forensic) and applications (both) for the quantification and analysis of content in video streams. The LVDBMS, along with the LVQL, links between multimedia content, streaming video analysis, and methods to provide contextual awareness. Comparisons between the LVDBMS and DBMS were previously shown in [14]. Here we present another application for DDDAS.

LVQL queries expressed in the LVDBMS are continuously evaluated and results streamed back to the corresponding user. The interval at which they are evaluated is referred to as the period of the query; (e.g., a query period of 1 second). The video dataset leveraged for the work was generated by the DARPA Video and Image Retrieval and Analysis Tool (VIRAT) program, which captured a number of activities using both aerial and ground-based video (e.g., three scenes of activities in Figures 11, 12, 13, numbered videos 1-3).

Complex and simple video clips were used to detect objects and extract activity analysis. Representative frames are shown in Fig. 11 for a simple scene showing an automobile and a group of people. Fig. 14 shows ground truth object counts and entrances/exits. The LVQL extracts metrics of activity and complexity from video as shown in Fig. 15.

Figure 11 – *Video1* - representative videos of people getting in car video.

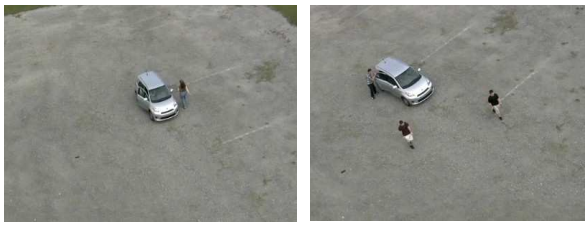

Figure 12 – *Video2* from the VIRAT collection depicting objects (people and vehicles), people and activities (walking, loading, unloading).

Figure 13 – *Video3* from the VIRAT collection depicting objects, people and activities (walking, shoveling, handing-off).

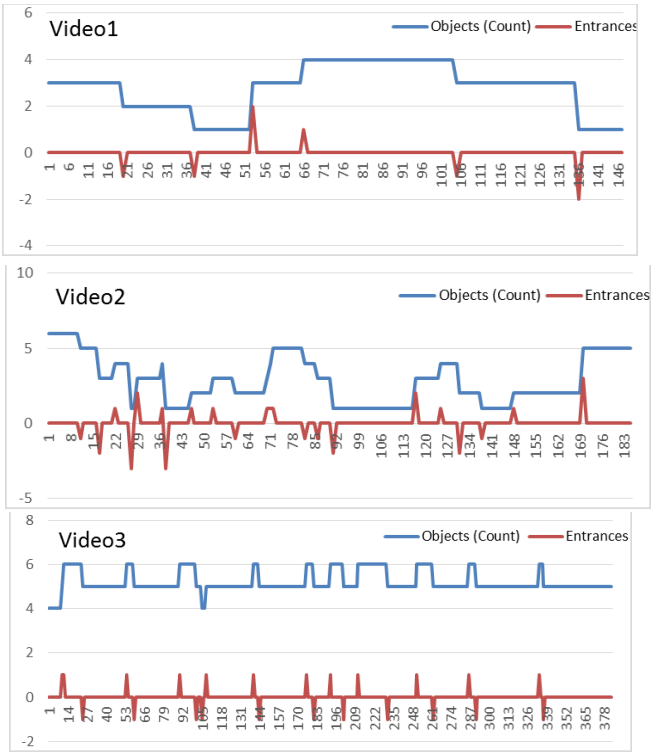

Figure 14 – Ground truth showing object count observed in a scene and corresponding derivative (change – either entrance or exit), for the video runtime measured in seconds.

Fig. 15 provides two metrics defined in terms of LVQL queries which help characterize the activity: labeled as "count" and "activity." *Count* is defined by the *query action 'appear' on appear(c0.*,400)*, which identifies when salient objects are observable and greater than or equal a certain size, in this case 250 pixels in area. The *Activity* metric is defined by the *query action 'appear ct' on count(appear(c0.*,400))*, which wraps the *appear* operator in an aggregate function *count* which cumulatively counts the number of distinct objects based upon its track. The derivative of the *count* metric provides the number of objects coming into or exiting the scene. Filtering over the activity count derivative indicates when activities are occurring that can be used to alert operators, cue other cameras, or determine scene-rich content. The LVQL query provides a concise summary of object entrance and exit activity in the video clip. Other LVQL spatial and temporal operators can produce extensions or refinements of this information to provide additional context such as activity in certain regions of the scene, and so forth.

The Merriam-Webster dictionary defines *veracity* as truthfulness, or conformity with truth or fact. To compare to "truth," related metrics presented in Table 4 provide corresponding information in terms of true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Assuming the video is objective, then the credible results are used. Table 5 gives metric averages computed over the video duration; which is the "quality" information such as precision, accuracy, and veracity (e.g., error) in the URREF ontology.

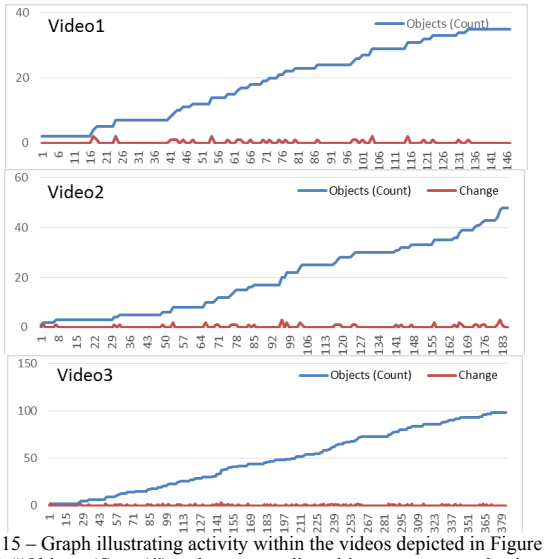

Figure 15 – Graph illustrating activity within the videos depicted in Figure 11-13 ("Objects (Count)") and corresponding object entrances and exits ("Change") number of objects identified ("Count"), for a portion of the video.

Table 4: Definition of metric equations

	Expert says yes	**Expert says no**
System says yes	*a* (TP)	*b* (FP)
System says no	*c* (FN)	*d* (TN)
Precision (*P*)= *a*/(*a*+*b*)	*Recall* (*R*) = *a*/(*a*+*c*)	
Error (*E*) = (*b*+*c*)/*n*	*Accuracy* (*A*) = (*a*+*d*)/*n*	

Table 5: URREF metrics over sample videos (E could be veracity)

Video	**P**	**R**	**A**	**E**
1	0.26	0.97	0.80	0.23
2	0.45	0.94	0.78	0.39
3	0.24	0.97	0.86	0.22
Average	*0.31*	*0.96*	*0.81*	*0.28*

From the analysis, *A* is the accuracy and the error (*E*) could be a measure of veracity in relation to whether the source assessment of what is true. The results correspond to all changes (e.g., entrance and exit). Further analysis of veracity will help to refine the URREF ontology.

VI. CONCLUSIONS

The LVDBMS, a stream-oriented database platform architected for real-time stream processing, enables content analysis via user-based queries, establishing Level 5 fusion. The LVDBMS facilitates timely responses to user queries by leveraging various hash and index structures that process and represent objects extracted from multimedia video streams. Queries expressed in LVQL syntax permit users and applications to specify events and thus enable multimedia fusion of query-based text to assist operators and analysts with scene and content understanding. Future efforts include using the information exploitation system for big-data problems including physics-based and human-based video-to-text information fusion [56]. Large scale multimedia applications will require dynamic-driven application systems approaches that bring together the context, privacy, and semantic-aware analysis.

ACKNOWLEDGEMENTS

The authors were supported under an AFOSR grant in Dynamic Data-Driven Applications Systems and support from AFRL. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U.S. Government.

REFERENCES

[1] B. Liu, E. Blasch, Y. Chen, A. J. Aved, *et al.*, "Information Fusion in a Cloud Computing Era: A Systems-Level Perspective," *IEEE Aerospace and Electronic Systems Mag.*, Vol. 29, No. 10, pp. 16 – 24, Oct. 2014.

[2] Z. Liu, E. Blasch, Z. Xue, R. Langaniere, and W. Wu, "Objective Assessment of Multiresolution Image Fusion Algorithms for Context Enhancement in Night Vision: A Comparative Survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(1):94-109, 2012.

[3] S. Ezekiel, M .G. Alford, *et al*. Multi-scale decomposition tool for Content Based Image Retrieval," *IEEE Applied Imagery Pattern Recognition Workshop*, 2013.

[4] X. Shi, H. Ling, *et al.,* "Context-Driven Moving Vehicle Detection in Wide Area Motion Imagery," *Int'l Conf on Pattern Recognition,* 2012.

[5] P. Liang, H. Ling, *et al*, "Vehicle Detection in Wide Area Aerial Surveillance using Temporal Context," *Int'l Conf. on Info Fusion*, 2013.

[6] E. Blasch, J. Garcia Herrero, L. Snidaro, J. Llinas, G. Seetharaman, K. Palaniappan, "Overview of contextual tracking approaches in information fusion," *Proc. SPIE*, Vol. 8747, 2013.

[7] E. Blasch, A. Steinberg, S. Das, J Llinas, C. Chong, O. Kessler, F. White,."Revisiting the JDL model for information Exploitation," *International Conference on Information Fusion*, 2013.

[8] E. Blasch, E. Bosse, D. A. Lambert, D. A., *High-Level Information Fusion Management and Systems Design*. Artech House, 2012.

[9] E. Blasch, J. Nagy, A. Aved, W. M. Pottenger, M. Schneider, R. Hammoud, E. K. Jones, A. Basharat, *et al*., "Context aided Video-to-Text Information Fusion," *Int'l.. Conf. on Information Fusion*, 2014.

[10] E. Blasch, J. Salerno, *et al.*, "Resource management coordination with level 2/3 fusion issues and challenges [Panel Report]," *Aerospace and Electronic Systems Magazine, IEEE*, *23*(3), 32–46, 2008/

[11] A. N. Steinberg, C. L. Bowman, *et al*., "Adaptive Context Assessment and Context Management," *Int'l.. Conf. on Information Fusion*, 2014.

[12] C. J. Date, *An Introduction to Database Systems* (2nd ed.). Addison-Wesley Publishing Company, Inc., 1977.

[13] P. C. G. Costa, K. B. Laskey, E. Blasch and A-L. Jousselme, "Towards Unbiased Evaluation of Uncertainty Reasoning: The URREF Ontology," *Int. Conf. on Info Fusion*, 2012.

[14] A. J. Aved, *Scene Understanding for Real Time Processing of Queries Over Big Data Streaming Video*. PhD, Univ. of Central Florida, 2013.

[15] E. Blasch, P. C. G. Costa, *et al*., "The URREF Ontology for Semantic Wide Area Motion Imagery Exploitation," *Int'l Conf. on Semantic Technologies for Intelligence, Defense, and Security* (STIDS), 2012.

[16] E. Blasch, K. B. Laskey, A-L. Jousselme, V. Dragos, P. C. G. Costa, and J. Dezert, "URREF Reliability versus Credibility in Information Fusion (STANAG 2511)," *Int'l Conf. on Info Fusion*, 2013.

[17] E. Blasch, A. Jøsang, J. Dezert, P. C. G. Costa, K. B. Laskey, A-L. Jousselme, "URREF Self-Confidence in Information Fusion Trust," *Int'l. Conf. on Information Fusion*, 2014.

[18] M .A. Tantaoui, K. A, Hua, T. T Do, BroadCatch: a periodic broadcast technique for heterogeneous video-on-demand. *IEEE Transactions on Broadcasting,* 50(3), 289–301, 2004.

[19] "Beyond 3V Issues - Veracity," Inside Big Data, 2013, accessed at http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data-veracity/

[20] E. Waltz, *Quantitative Intelligence Analysis*, Rowman& Littlefield, 2014.

[21] D. K. B. Ismail, O. Grivard, "A model-driven approach to the a priori estimation of operator workload," *CogSIMA*, 2015.

[22] I. Claverie-Berge, "Solutions Big Data IBM.," 2012. Accessed at: http://www05.ibm.com/fr/events/netezzaDM_2012/ Solutions_Big_Data.pdf

[23] F. Darema, DDDAS Workshop Groups. *Creating a dynamic and symbiotic coupling of application/ simulations with measurements/experiments*. NSF DDDAS Workshop, 2000. Available via http://www.nsf.gov/cise/cns/dddas/ [accessed Jan 2013].

[24] M. P. Hunter, R. M. Fujimoto, W. Suh, "An investigation of real-time Dynamic Data Driven transportation," *Proceedings of the IEEE Winter Simulation Conference*, 2006.

[25] L. Fan, L. Xiong, V. S. Sunderam, "Differentially private multi-dimensional time series release for traffic monitoring," *IFIP Data and Applications Security and Privacy Conf*., 2013.

[26] E. Blasch, "Enhanced Air Operations Using JView for an Air-Ground Fused Situation Awareness UDOP," *AIAA/IEEE Digital Avionics Systems Conference*, Syracuse, NY, Oct. 2013.

[27] E. Blasch, A. J. Aved, "Dynamic Data-Driven Application System (DDDAS) for Video Surveillance User Support," *International Conference on Computational Science*, 2015.

[28] H. Samet, *The design and analysis of spatial data structures* (Vol. 85). Addison-Wesley Reading MA, 1990.

[29] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[30] R. M. Bolle, B. L. Yeo, M. Yeung, "Video query: Research directions," *IBM Journal of Research and Development*, *42*(2), 233–252, 1998.

[31] R. Brunelli, O. Mich, C. M. Modena, "A Survey on the Automatic Indexing of Video Data," *J. of Visual Comm. and Image Representation*, *10*(2), 78–112, 1999.

[32] Y. Wang, Z. Liu, J. C. Huang, "Multimedia content analysis-using both audio and visual clues," *IEEE Signal Proc. Mag.,* 17(6), 12–36, 2000.

[33] C. G. M. Snoek, M. Worring, "Multimodal video indexing: A review of the state-of-the-art," *Multimedia Tools and App.*, *25*(1), 5–35, 2005.

[34] J. M. Boggs, *The art of watching films*. ERIC, 1996.

[35] R. Jain, A. Hampapur, "Metadata in video databases," *ACM Sigmod Record*, *23*(4), 27–33, 1984.

[36] P. Belhumeur, J. Hespanha, D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *European Conference on Computer Vision*, 43–5, 19996.

[37] A. K. Jain, R. Duin, *et al*., "Statistical pattern recognition: A review," *IEEE Tr. on Pattern Analysis and Machine Int., 22*(1), 4–37, 2000.

[38] I. Ide, K. Yamamoto, H. Tanaka, "Automatic video indexing based on shot classification," *Adv. Multimedia Content Processing*, 87–10, 1999.

[39] A. Nagasaka, Y. Tanaka, "Automatic video indexing and full-video search for object appearances," *Proc. IFIP*, pp. 113-127, 1992.

[40] E. Wold, T., Blum, D., Keislar, J. Wheaten, J. "Content-based classification, search, and retrieval of audio," *MultiMedia, IEEE*, *3*(3), 27–36, 1996.

[41] J. Foote, "Content-based retrieval of music and audio," *Proc. SPIE*, Vol. 3229, 1997.

[42] J. Nievergelt, H. Hinterberger, K. C Sevcik, "The grid file: An adaptable, symmetric multikey file structure," *ACM Transactions on Database Systems (TODS)*, *9*(1), 38–71, 1984.

[43] A. V. Aho, J. E. Hopcroft, J. Ullman, J. *Data structures and algorithms*. Addison-Wesley Longman Publishing Co., Inc, 1983.

[44] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms*. MIT press, 2001.

[45] J. Pieprzyk, B. Sadeghiyan, *Design of hashing algorithms*. Springer-Verlag New York, Inc., 2001.

[46] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, *18*(9), 509–51, 1975.

[47] P. Scheuermann, M. Ouksel, "Multidimensional B-trees for associative searching in database systems," *Information Sys.*, *7*(2), 123–137, 1982.

[48] W. Aref, I. F. Ilyas, "Sp-gist: An extensible database index for supporting space partitioning trees," *Journal of Intelligent Information Systems*, *17*(2), 215–240, 2001.

[49] A. Guttman, *R-trees: a dynamic index structure for spatial searching* (Vol. 14). ACM, 1984.

[50] G. Hristescu, M. Farach-Colton, *Cluster-preserving embedding of proteins*. Tech. Report 99-50, Comp. Sci. Depart., Rutgers Univ., 1999.

[51] J. T. L. Wang, X. Wang, K. I.. Lin, D. Shasha, *et al*., "Evaluating a class of distance-mapping algorithms for data mining and clustering," *ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, 1999.

[52] T. Lukoianova, V. Rubin, "Veracity Roadmap: Is Big Data Objective, Truthful and Credible?," *Advances In Classification Research Online*, 24(1), 2014. doi:10.7152/acro.v24i1.14671.

[53] G. Bradski, The OpenCV Library. *Dr. Dobb's Journal of Software Tool,* 2000.

[54] S. Taylor, *Optimizing Applications for Multi-Core Processors, Using the Intel Integrated Performance Primitives*, Intel Press, 2007.

[55] E. Blasch, Z. Wang, H. Ling, K. Palaniappan, G. Chen, D. Shen, A. Aved, *et al*., "Video-Based Activity Analysis Using the L1 tracker on VIRAT data," *IEEE Applied Imagery Pattern Rec. Workshop*, 2013.

[56] R. I. Hammoud, C. S. Sahin, *et al*., "Automatic Association of Chats and Video Tracks for Activity Learning and Recognition in Aerial Video Surveillance," *Sensors*, 14, 19843-19860, 2014.